СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ПРИ ДОСЛІДЖЕННІ І ПРОЕКТУВАННІ МЕХАНІЗМІВ ВИЯВЛЕННЯ АТАК В ОПЕРАЦІЙНІЙ СИСТЕМІ, ЯКІ ОСНОВАНІ НА ІНТЕЛЕКТУАЛЬНИХ СИСТЕМАХ

О.М. Бажанюк
Національний технічний університет України
«Київський політехнічний інститут» Фізико-технічний інститут
e-mail: virvdova@gmail.com

Нейроні мережі, генетичні алгоритми, нейродинаміка є актуальними механізмами обробки інформації, але вони не використовуються в системах виявлення атак. В більшості з них використовуються класичні статистичні механізми. Проблема підвищення продуктивності систем виявлення атак є актуальною. Ще одна проблема яка є актуальною - виявлення вразливостей коду програмного забезпечення. В даній області проводять багато досліджень, наприклад National ICT Australia (NICTA) – створює математичні моделі для систем будь-якої складності, в тому числі і для ОС, Coverity (*www.coverity.com*) – створює систему виявлення вразливостей коду, використовуючи статистичну обробку даних, checkmycode.org - це готовий аналізатор програмного коду, який виявляє логічні помилки.

Для прикладу системи програмного забезпечення будемо використовувати ядро відкритої операційної системи Linux версії 2.6.30-R6. Для підвищення продуктивності ядра були спроби спроектувати планувальник, який використовує евристичні алгоритми. Проте він програє у продуктивності планувальникам, які використовують просту математику. Але актуальність впровадження нейронних мереж в інші механізми ядра залишається. Проаналізуємо способи використання нейронних мереж в ядрі ОС.
Є кілька варіантів:

Нейронна система, яка буде аналізувати початкові тексти ядра і знаходити вразливості. Для такого аналізу нам необхідна наявність послідовності початкових кодів ядер та файлів, в яких були виявленні помилки. Код групується за функціями, файлами, зв’язками. Вибирається такий код, логіка якого була як мінімум в 25% ядер. Початкові тексти розкладається за допомогою граматик в синтаксичне дерево [1-3]. Потім ці дані є навчальною вибіркою для нейронної мережі при методі навчання з учителем. У нейронній мережі задаються операції з отриманими структурами (синтаксичними деревами) і, можливо, складається група з цих елементів. За допомогою такої мережі ми зможемо аналізувати помилки і їх прояви в наступних версіях ядра. Проблеми, які можуть виникнути - це складність обчислення, негарантований результат, помилки не є статистично очікуваними. Також існує строгий доказ безпомилковості коду для мікроядра (Secure мікроядра (seL4)). Дана теорія була представлена NICTA. У цьому доказі, є обмеження на помилки пам'яті, BIOS. Теорія доводиться за допомогу комплексу автоматичного математичного докази Isabelle / HOL [13,14]. Дана теорія зображає початкові коди ядра у вигляді математичної абстракції. Далі задається правила і аксіоми для виявлення помилкових кодів. На останньому етапі доводиться справедливість теорії за допомогою Isabelle / HOL.

Розглянемо другий варіант: нейронна система аналізує ядро, яке компілюється. При компілювання ядра, система автоматично складає статистику значень змінних функції та самого компілятора. Дані значення будуть навчальною вибіркою для нейронної мережі. Для складання статистики використовувати GDB [4-7], strace. Недоліки - складність

обчислень.

Третій варіант на практиці найпростіший. Нейронна система, яка аналізує діюче ядро на предмет атак. Саме це і є основна практична частина моєї роботи. Система підтримки прийняття рішень спроектована на результатах роботі нейронної мережі. Вона приймає наступні рішення: блокувати атаку, повідомити про атаку або ігнорувати данні.

Отримані висновки:

Спроектовано систему підтримки прийняття рішень.

Запропоновано аналіз одного з механізмів діючого ядра операційної системи Linux на наявність атак. Метод аналізу оснований на алгоритмах нейронних мереж. Нейронні мережі: одно- та багатошарові персептрони. Отримані результати підтверджують складність виявлення залежності значень параметрів ядра. Помилка в експериментах ~ 17%.

При більш глибокому аналізі можлива побудова системи підтримки прийняття рішень на всіх рівнях і у всіх механізмах ядра. Програма подальших досліджень буде включати в себе розширення діапазону методів статистичного аналізу вхідних даних, ускладнення структури нейронної мережі, застосування нейронних мереж з пам'яттю, методи нейродинаміки. А також збільшення кількості даних, наприклад додавання даних з механізмів мьютексом (mutex) і потоків (threads).

Література

1. Linux Kernel Organization (1994) The Linux Kernel Archives http://www.kernel.org/pub/linux/kernel/

2. Suzette Pelouch (1998) Java Language Specification http://java.sun.com/docs/books/jls/first_edition/html/19.doc.html

3. Ralf Lämmel (2005) Engineering of Grammarware http://www.cs.vu.nl/grammars/

4. http://en.wikipedia.org/wiki/Category:Static_code_analysis

5. http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

6. Free Software Foundation (2009) GDB: The GNU Project Debugger http://www.gnu.org/software/gdb/

7. Yves Younan, Wouter Joosen, Frank Piessens.(July 2004) Code Injection in C and C++ : A Survey of Vulnerabilities and Countermeasures, Report CW386, Katholieke Universiteit Leuven, Department of Computer Science

8. Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch and Simon Winwood (2009) Secure Microkernel Project (seL4) http://ertos.nicta.com.au/research/sel4/

9. Larry Paulson, Tobias Nipkow (2009) Isabelle _http://isabelle.in.tum.de

10. Arne Georg Gleditsch (1995) Linux Cross Reference http://lxr.linux.no

11. M. Tim Jones (2007) Anatomy of Linux synchronization methods http://www.ibm.com/developerworks/linux/library/l-linux-synchronization.html